A Transport Protocol for Content-Based Publish/Subscribe Networks

Amir Malekpour

University of Lugano, Faculty of Informatics
Advisors: Antonio Carzaniga, Fernando Pedone

Outline

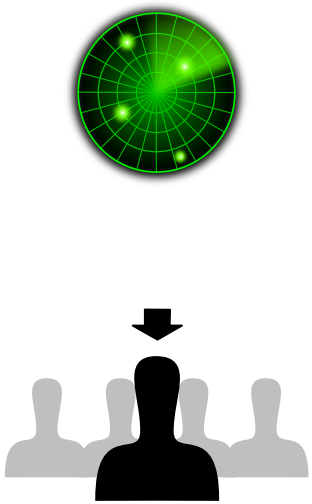
- Publish-Subscribe communication paradigm
  - ▷ Content-based
  - ▷ Best-effort
- Problem and solution outline
- Transport Protocol
  - ▷ FIFO ordering
  - ▷ Reliable delivery

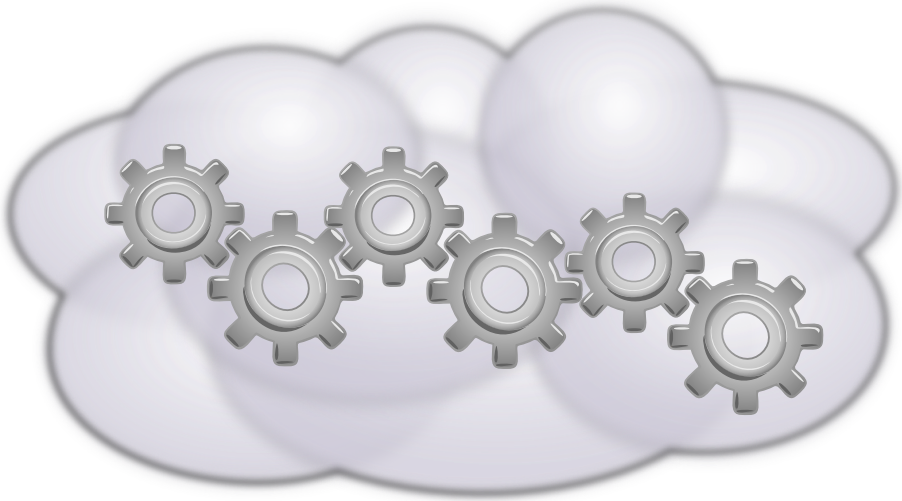
Publish/Subscribe Network

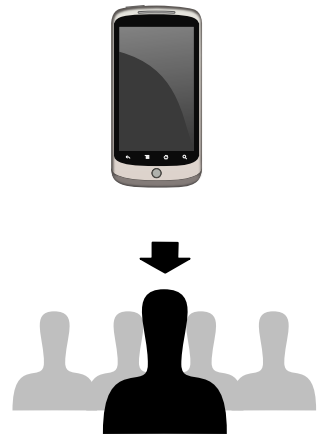
Publishers

Event processing/dissemination

Subscribers





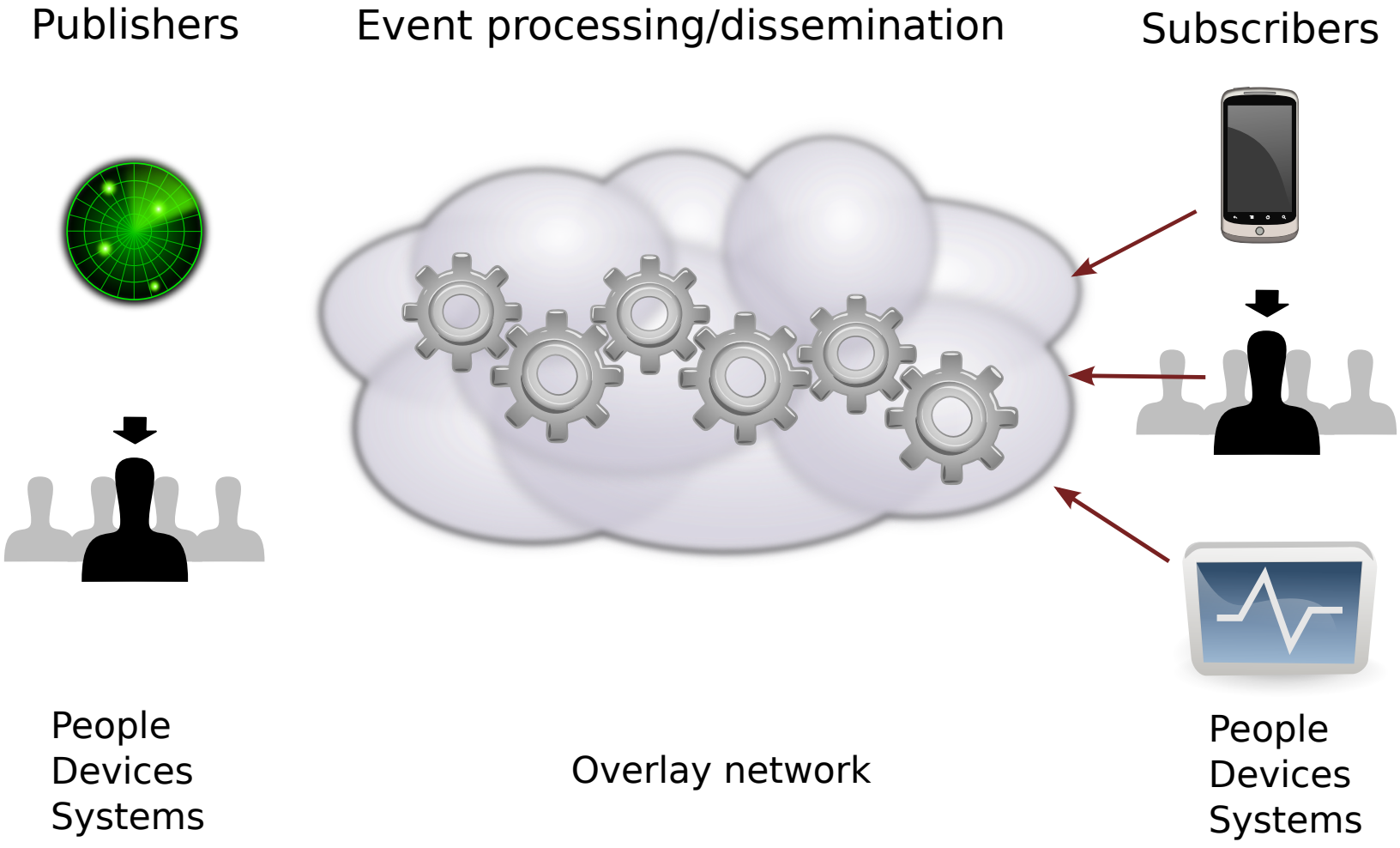


People
Devices
Systems

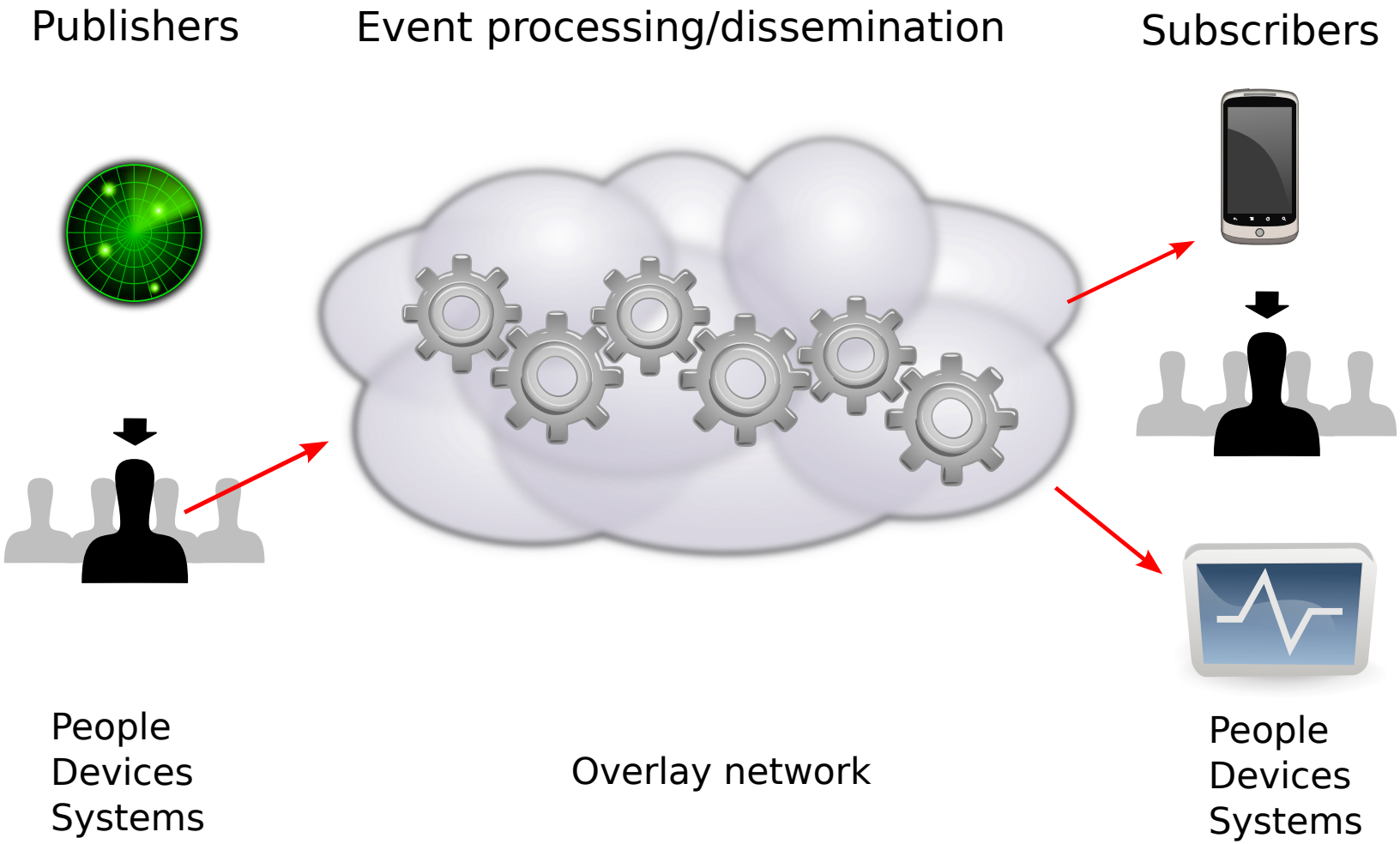
Overlay network

People
Devices
Systems

Publish/Subscribe Network



Publish/Subscribe Network



Events

- RFID events 4 trillion events per day

Source: G. T. Lakshmanan et al., A stratified approach for supporting high throughput event processing applications, DEBS 2009

Events

- RFID events 4 trillion events per day
- Financial applications in large banks 400 million events per day

Source: G. T. Lakshmanan et al., A stratified approach for supporting high throughput event processing applications, DEBS 2009

Events

- RFID events 4 trillion events per day
- Financial applications in large banks 400 million events per day
- Massively multiplayer online games (MMOG)millions of events per second during peak periods.

Source: G. T. Lakshmanan et al., A stratified approach for supporting high throughput event processing applications, DEBS 2009

Publish/Subscribe

- Distriuted, asynchronous, large scale service
- A set of algorithms and potocols for routing events

Publish/Subscribe

- Distriuted, asynchronous, large scale service
- A set of algorithms and potocols for routing events

Architectures

There is a plethora of different designs and architectures

Publish/Subscribe

- Distriuted, asynchronous, large scale service
- A set of algorithms and potocols for routing events

Architectures

There is a plethora of different designs and architectures

Standards

- Java Messaging Service (JMS)
- Advanced Message Queuing Protocol (AMQP)
- Data Distribution Service (DDS)

Classifying Existing Systems

Categorizing based on many factors such as:

- Event model: topic-based, **content-based**
- Ordering and delivery guarantees: guaranteed, **best-effort**

Classifying Existing Systems

Categorizing based on many factors such as:

- Event model: topic-based, **content-based**
- Ordering and delivery guarantees: guaranteed, **best-effort**

The focus of thesis is **content-based, best-effort** networks

Event Model

Content-based:

- An event consists of a tuple of attribute name/value

  (class = “stock”, symbol = “IBM”, price = 125)

Event Model

Content-based:

- An event consists of a tuple of attribute name/value

  (class = "stock", symbol = "IBM", price = 125)

- Subscriptions are a logical combination of constraints on attribute name/value:

  ((class = "sport") $\wedge$ (country = "UK") $\wedge$ (club = "Liverpool")) $\vee$ ((class = "stocks") $\wedge$ (symbol = "IBM") $\wedge$ (price $<$ 120))

Event Model

Content-based:

- An event consists of a tuple of attribute name/value

  (class = "stock", symbol = "IBM", price = 125)

- Subscriptions are a logical combination of constraints on attribute name/value:

  ((class = "sport") ∧ (country = "UK") ∧ (club = "Liverpool")) ∨ ((class = "stocks") ∧ (symbol = "IBM") ∧ (price < 120))

- **Fine grained filtering**
  - ▷ Numeric operators: $< = >$
  - ▷ String operators: substring, prefix, suffix

Event Model

Content-based:

- An event consists of a tuple of attribute name/value

  (class = "stock", symbol = "IBM", price = 125)

- Subscriptions are a logical combination of constraints on attribute name/value:

  ((class = "sport") ∧ (country = "UK") ∧ (club = "Liverpool")) ∨ ((class = "stocks") ∧ (symbol = "IBM") ∧ (price < 120))

- **Fine grained filtering**
  - ▷ Numeric operators: $< = >$
  - ▷ String operators: substring, prefix, suffix

Guaranteed Service vs. Best-effort

Guaranteed delivery and FIFO ordering

- Necessary for some applications
- Costly at run-time: higher end-to-end delay, lower throughput
- Apache ActiveMQ

Guaranteed Service vs. Best-effort

Guaranteed delivery and FIFO ordering

- Necessary for some applications
- Costly at run-time: higher end-to-end delay, lower throughput
- Apache ActiveMQ

Best-effort systems

- Target high throughput and low end-to-end delay
  - ▷ No "store-and-forward"
  - ▷ No acknowledgement

Guaranteed Service vs. Best-effort

Guaranteed delivery and FIFO ordering

- Necessary for some applications
- Costly at run-time: higher end-to-end delay, lower throughput
- Apache ActiveMQ

Best-effort systems

- Target high throughput and low end-to-end delay
  - ▷ No "store-and-forward"
  - ▷ No acknowledgement
- Less complex
  - ▷ More optimizations
  - ▷ Hardware-assisted implementation feasible

Guaranteed Service vs. Best-effort

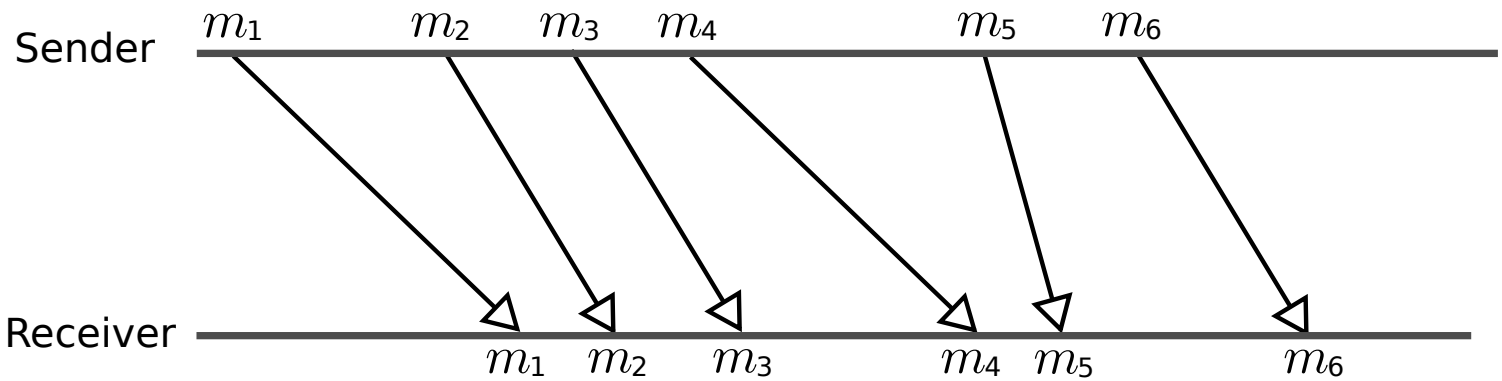
Guaranteed delivery and FIFO ordering

- Necessary for some applications
- Costly at run-time: higher end-to-end delay, lower throughput
- Apache ActiveMQ

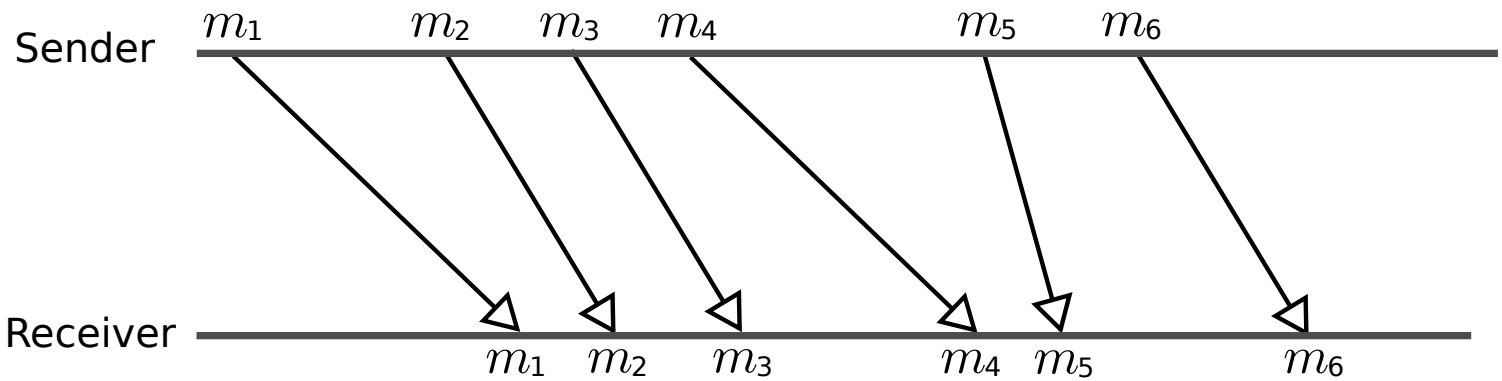
Best-effort systems

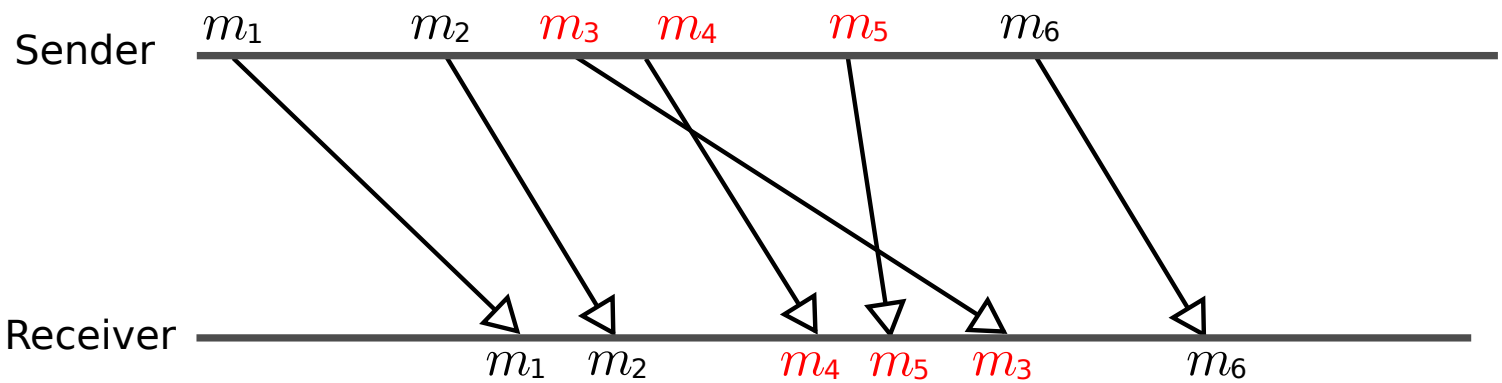
- Target high throughput and low end-to-end delay
  - ▷ No "store-and-forward"
  - ▷ No acknowledgement
- Less complex
  - ▷ More optimizations
  - ▷ Hardware-assisted implementation feasible
- Datagram-like messages
  - ▷ Out-of-order deliveries
  - ▷ Message loss
- Siena B-DRP

FIFO Ordering

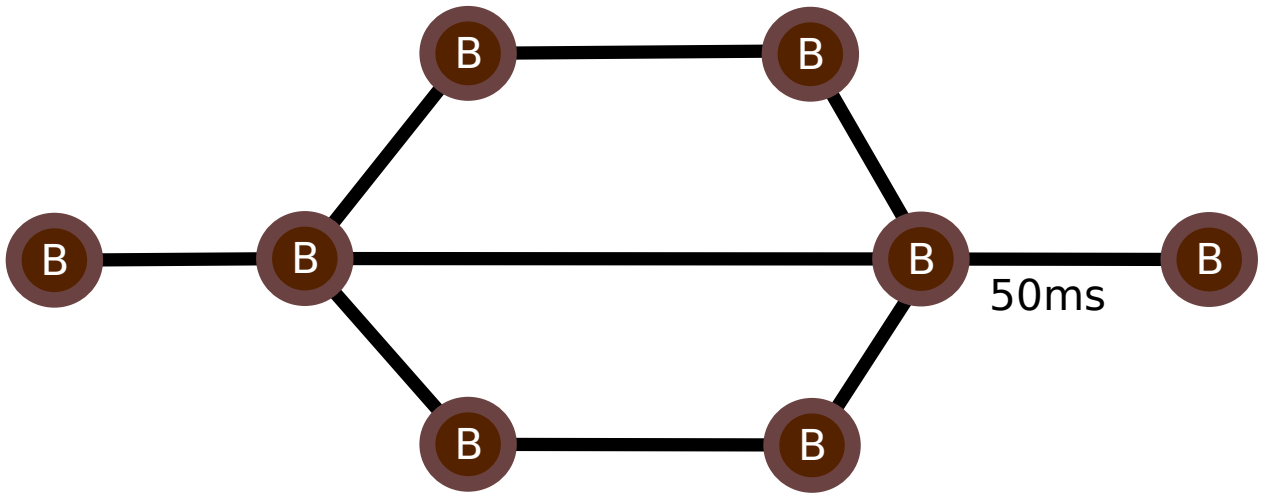


FIFO Ordering



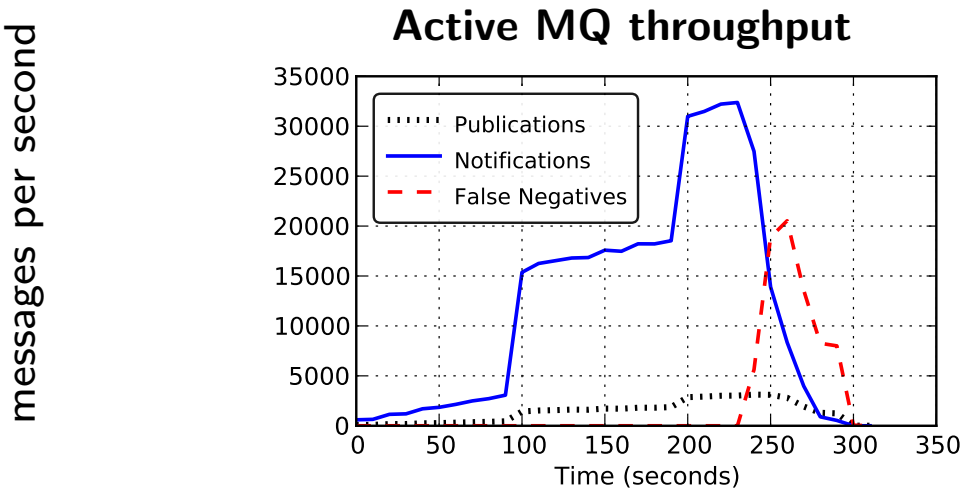


Experiment Setup

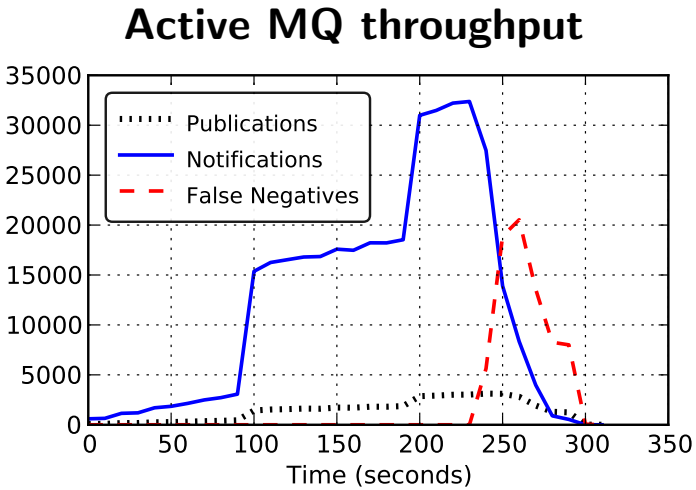


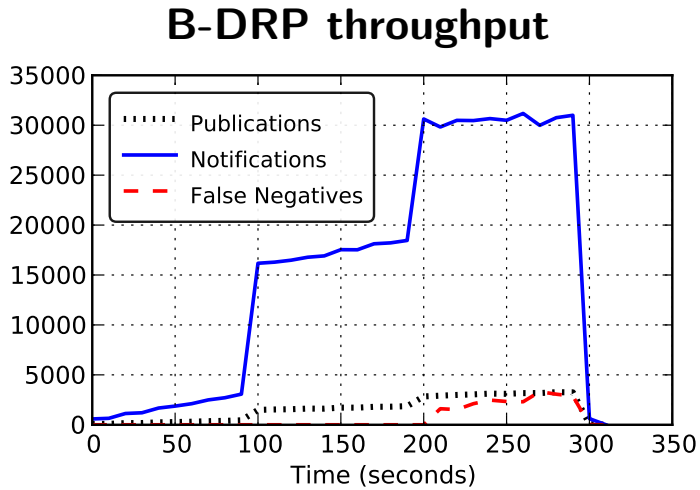
- 8-brokers
- Each broker serves 50 clients
- 50 publishers, 350 subscribers

ActiveMQ vs. B-DRP



ActiveMQ vs. B-DRP

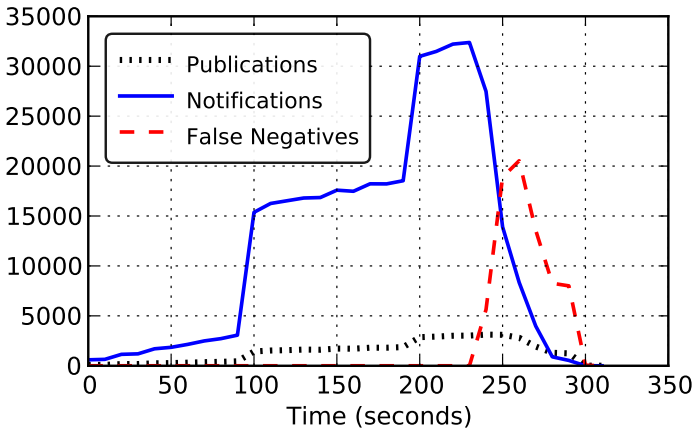




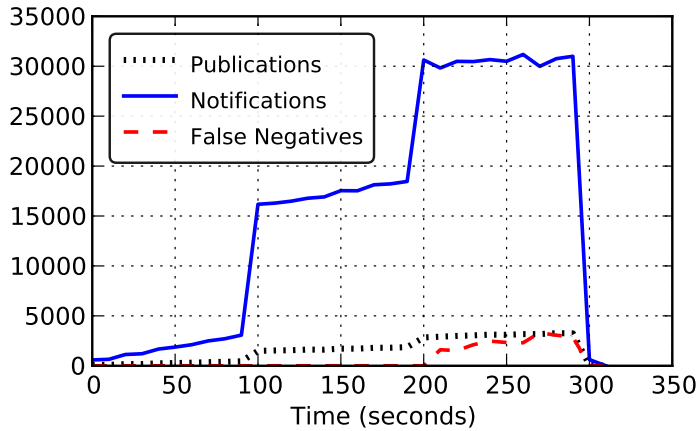
ActiveMQ vs. B-DRP

messages per second

Active MQ throughput



B-DRP throughput



messages per second

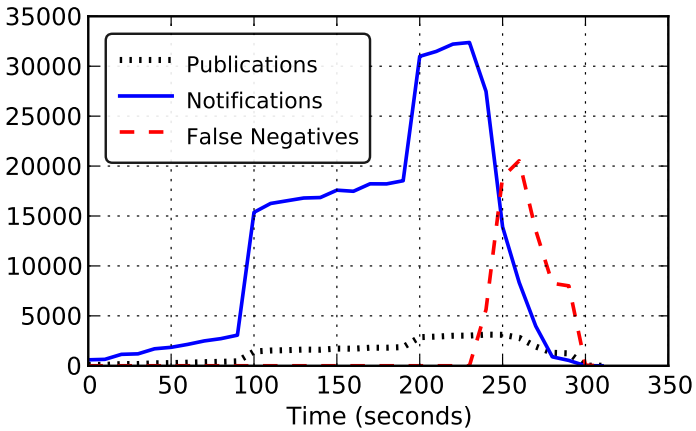
Active MQ FIFO violations

0

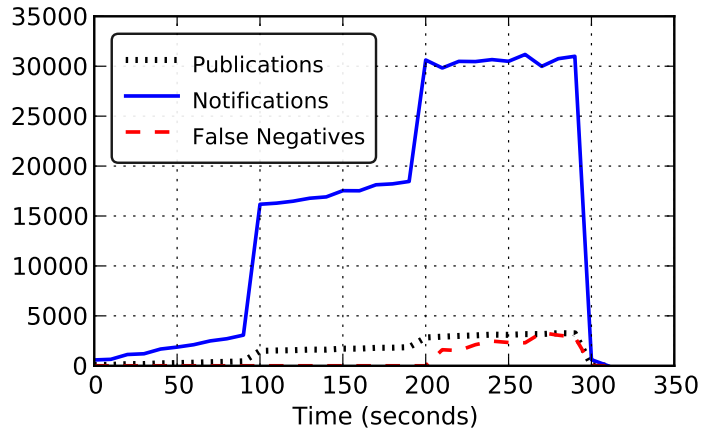
ActiveMQ vs. B-DRP

messages per second

Active MQ throughput



B-DRP throughput

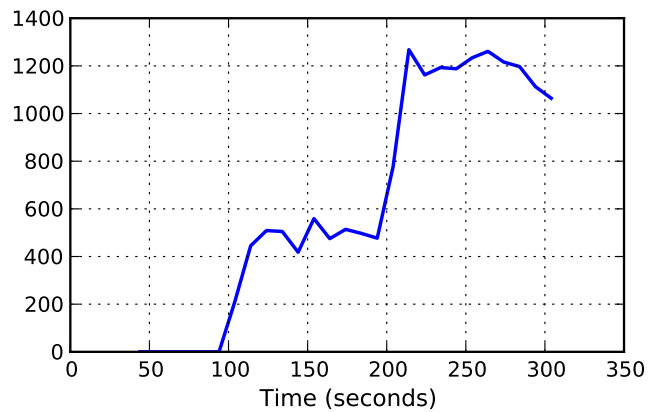


messages per second

Active MQ FIFO violations

0

B-DRP FIFO violations



Nearly 173,000 FIFO violations

Proposal

Like always...

There is a tension between desirable factors e.g., throughput vs. ordering

Proposal

Like always...

There is a tension between desirable factors e.g., throughput vs. ordering

- Can we mitigate this tension in best-effort systems?

Proposal

Like always...

There is a tension between desirable factors e.g., throughput vs. ordering

- Can we mitigate this tension in best-effort systems?

Proposal

Like always...

There is a tension between desirable factors e.g., throughput vs. ordering

- Can we mitigate this tension in best-effort systems?
  - ▷ Not at the protocol/broker level

Proposal

Like always...

There is a tension between desirable factors e.g., throughput vs. ordering

- Can we mitigate this tension in best-effort systems?
  - ▷ Not at the protocol/broker level
  - ▷ Look at the broker network as a black box

Proposal

Like always...

There is a tension between desirable factors e.g., throughput vs. ordering

- Can we mitigate this tension in best-effort systems?
  - ▷ Not at the protocol/broker level
  - ▷ Look at the broker network as a black box
- **End-to-end "transport protocol"**

Transport Protocol

- Not new, we've been using them since the 70s (TCP)

Transport Protocol

- Not new, we've been using them since the 70s (TCP)
- On top of a datagram (best-effort) protocol e.g.: IP

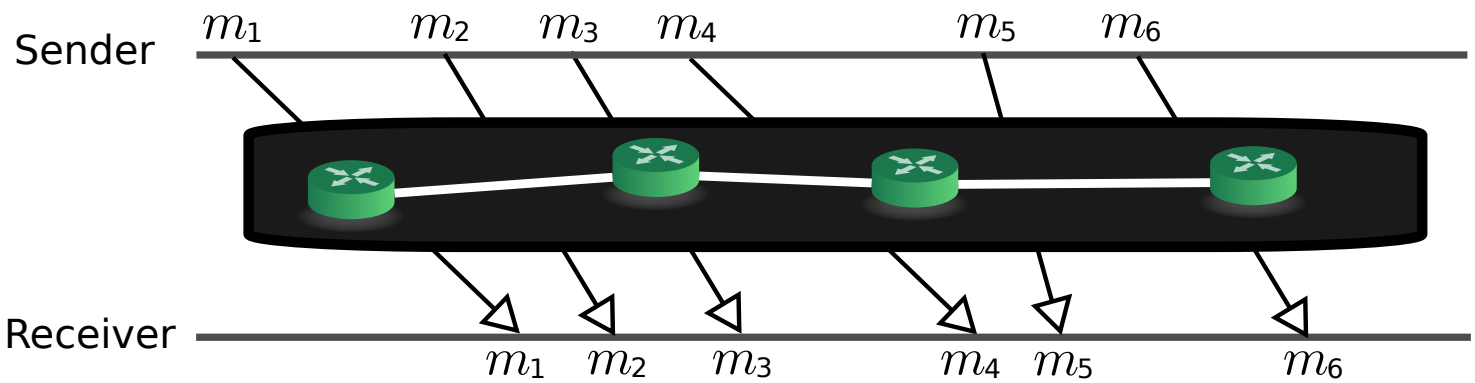
Transport Protocol

- Not new, we've been using them since the 70s (TCP)
- On top of a datagram (best-effort) protocol e.g.: IP
- Three main purposes:
  - **Message ordering**
  - **Guaranteed delivery**
  - Congestion control

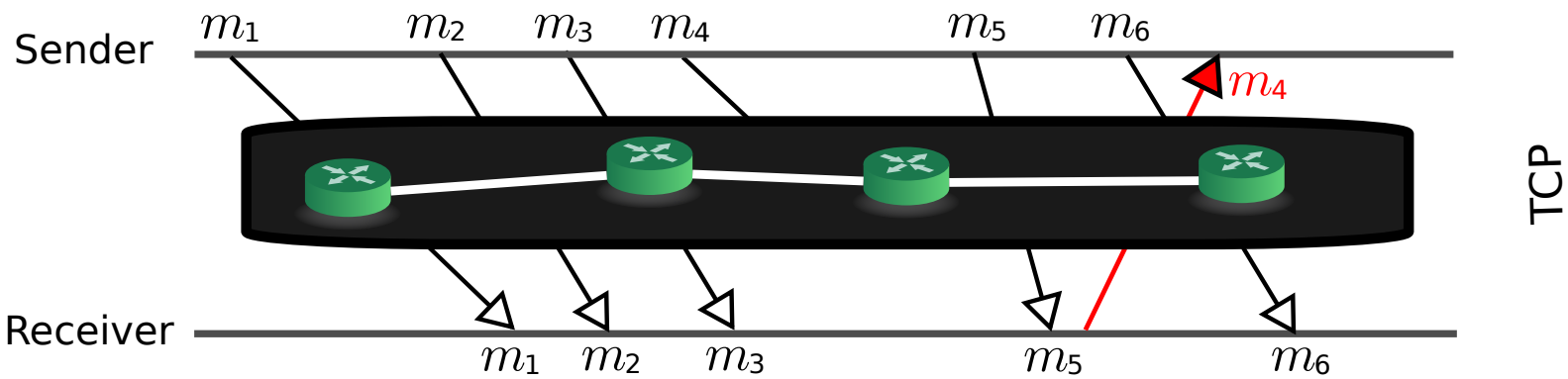
Transport Protocol

- Not new, we've been using them since the 70s (TCP)
- On top of a datagram (best-effort) protocol e.g.: IP
- Three main purposes:
  - **Message ordering**
  - **Guaranteed delivery**
  - Congestion control
- But there are some fundamental differences...

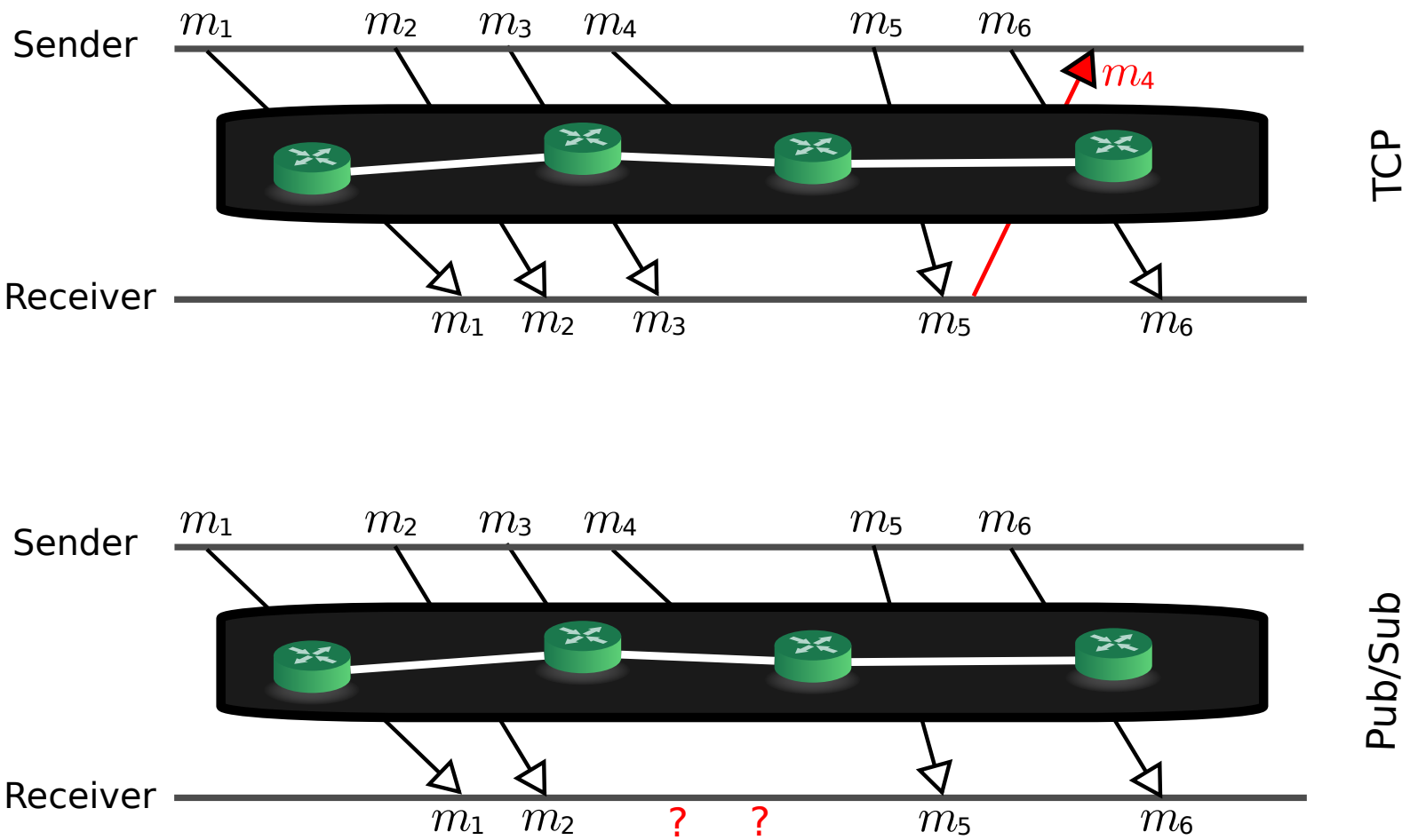
Sequence Numbers



Sequence Numbers



Sequence Numbers



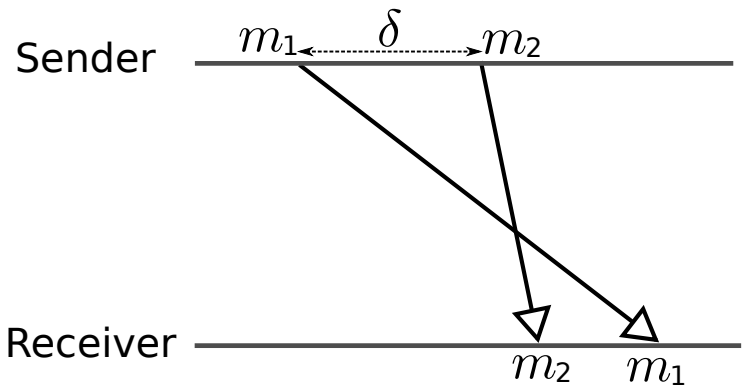
FIFO Ordering

FIFO Violation

- What causes FIFO violation?

FIFO Violation

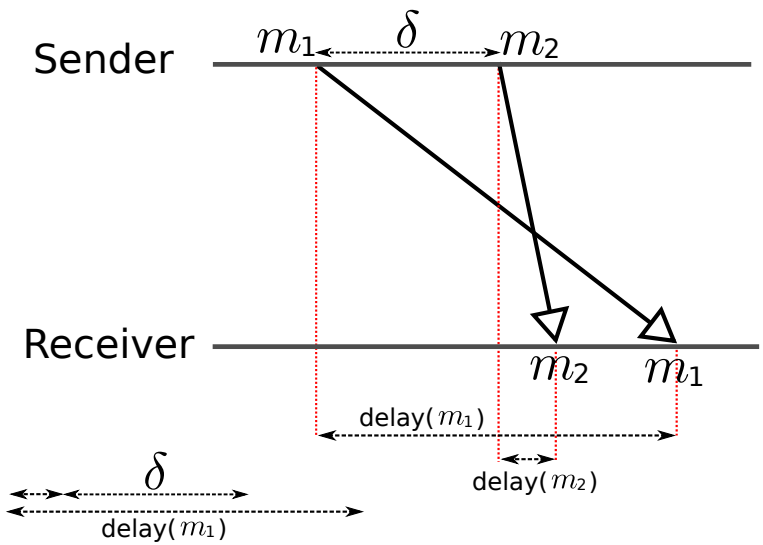
- What causes FIFO violation?



$$delay(m_1) > delay(m_2) + \delta$$

FIFO Violation

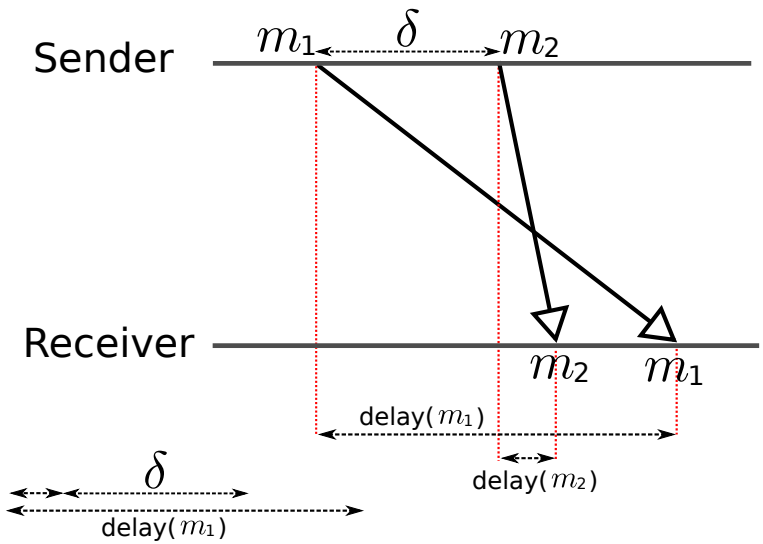
- What causes FIFO violation?



$$delay(m_1) > delay(m_2) + \delta$$

FIFO Violation

- What causes FIFO violation?

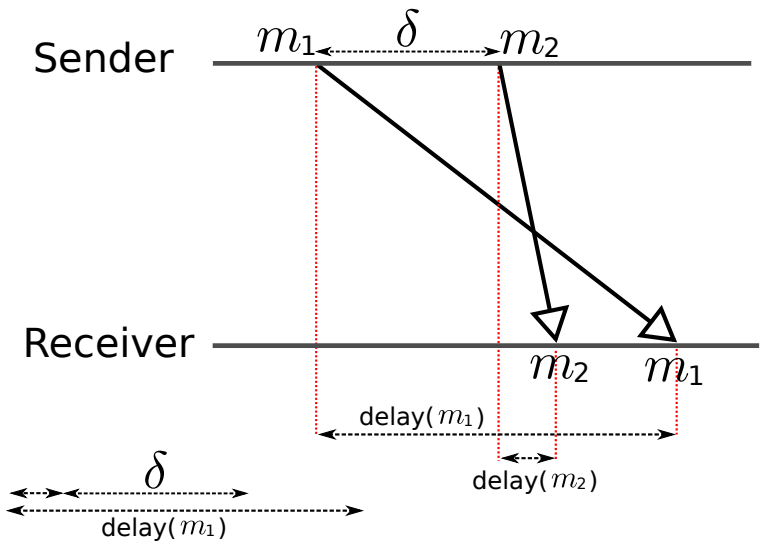


$$delay(m_1) > delay(m_2) + \delta$$

$$\Pr(\text{FIFO violation}) = \Pr(delay(m_1) - delay(m_2) > \delta)$$

FIFO Violation

- What causes FIFO violation?



$$delay(m_1) > delay(m_2) + \delta$$

$$\Pr(\text{FIFO violation}) = \Pr(delay(m_1) - delay(m_2) > \delta)$$

- $\underbrace{delay(m_1) - delay(m_2)}$ $\sim$?

Delay Differences

- Observed that messages published close to each other get "swapped"

Delay Differences

- Observed that messages published close to each other get "swapped"
  - ▷ $delay(m) = \underbrace{delay^*(m)}_{\text{constant}} + \underbrace{vardelay(m)}_{\text{random variable}}$
  - ▷ $delay^*(m)$ long-term-average link, queuing, and processing delays
  - ▷ $vardelay(m)$ short-term-variable delay

Delay Differences

- Observed that messages published close to each other get "swapped"
  - ▷ $delay(m) = \underbrace{delay^*(m)}_{\text{constant}} + \underbrace{vardelay(m)}_{\text{random variable}}$
  - ▷ $delay^*(m)$ long-term-average link, queuing, and processing delays
  - ▷ $vardelay(m)$ short-term-variable delay
- We assume that $delay^*(m_1) = delay^*(m_2)$

Delay Differences

- Observed that messages published close to each other get "swapped"
  - ▷ $delay(m) = \underbrace{delay^*(m)}_{\text{constant}} + \underbrace{vardelay(m)}_{\text{random variable}}$
  - ▷ $delay^*(m)$ long-term-average link, queuing, and processing delays
  - ▷ $vardelay(m)$ short-term-variable delay
- We assume that $delay^*(m_1) = delay^*(m_2)$
- $delay(m_1) - delay(m_2) \sim (X_1 - Y_1) + (X_2 - Y_2) + \cdots + (X_k - Y_k)$
  - ▷ X_i and Y_i is the processing time of message at broker i

Delay Differences

- Observed that messages published close to each other get "swapped"
  - ▷ $delay(m) = \underbrace{delay^*(m)}_{\text{constant}} + \underbrace{vardelay(m)}_{\text{random variable}}$
  - ▷ $delay^*(m)$ long-term-average link, queuing, and processing delays
  - ▷ $vardelay(m)$ short-term-variable delay
- We assume that $delay^*(m_1) = delay^*(m_2)$
- $delay(m_1) - delay(m_2) \sim (X_1 - Y_1) + (X_2 - Y_2) + \cdots + (X_k - Y_k)$
  - ▷ X_i and Y_i is the processing time of message at broker i
- Probabilistic bounds can be found on this random variable (e.g. Bernstein inequality)
- In case of Siena B-DRP a Hypoexponential distribution fits the random variable

Distribution Parameters

- How does the recipient estimate the parameters?

Distribution Parameters

- How does the recipient estimate the parameters?
  - With a little help from the sender

Distribution Parameters

- How does the recipient estimate the parameters?
  - With a little help from the sender
  - Sender piggybacks each message with its send time $departure(m)$

Distribution Parameters

- How does the recipient estimate the parameters?
  - With a little help from the sender
  - Sender piggybacks each message with its send time $departure(m)$
- Recipient records reception time of each message $arrival(m)$

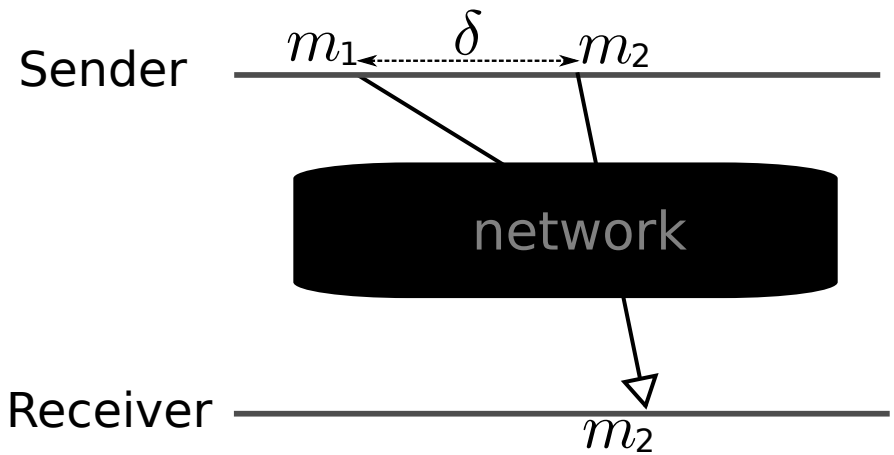
Distribution Parameters

- How does the recipient estimate the parameters?
  - With a little help from the sender
  - Sender piggybacks each message with its send time $departure(m)$
- Recipient records reception time of each message $arrival(m)$
- $x_i = delay(m_i) - delay(m_j) =$
  $[arrival(m_i) - arrival(m_j)] - [departure(m_i) - departure(m_j)]$

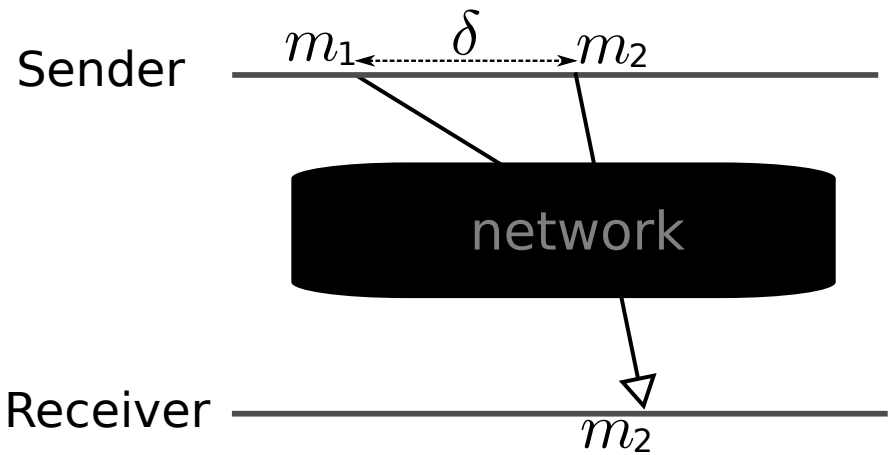
Distribution Parameters

- How does the recipient estimate the parameters?
  - With a little help from the sender
  - Sender piggybacks each message with its send time $departure(m)$
- Recipient records reception time of each message $arrival(m)$
- $x_i = delay(m_i) - delay(m_j) =$
  $[arrival(m_i) - arrival(m_j)] - [departure(m_i) - departure(m_j)]$
- Clock differences cancel out

Latch Time

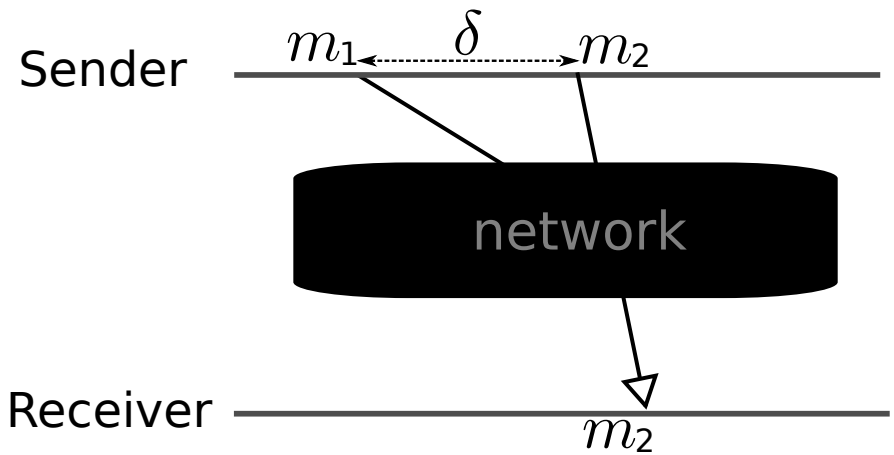


Latch Time



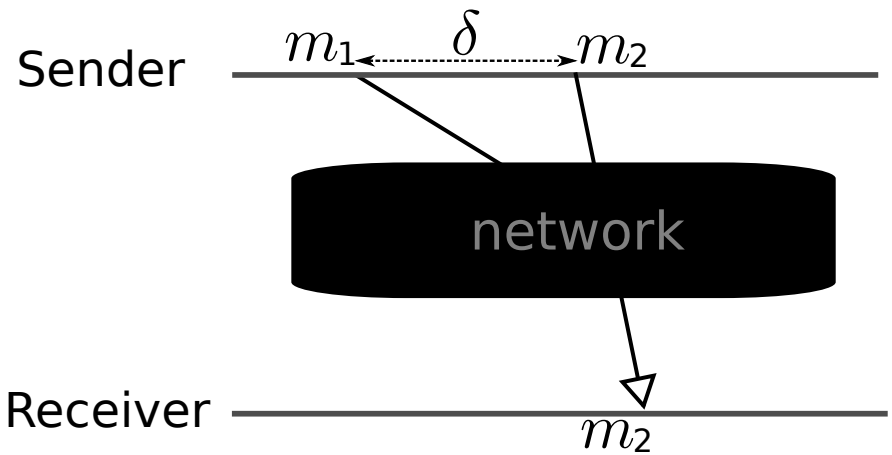
- Is m_1 relevant (matching) ?

Latch Time



- Is m_1 relevant (matching) ?
- Let's assume it is...

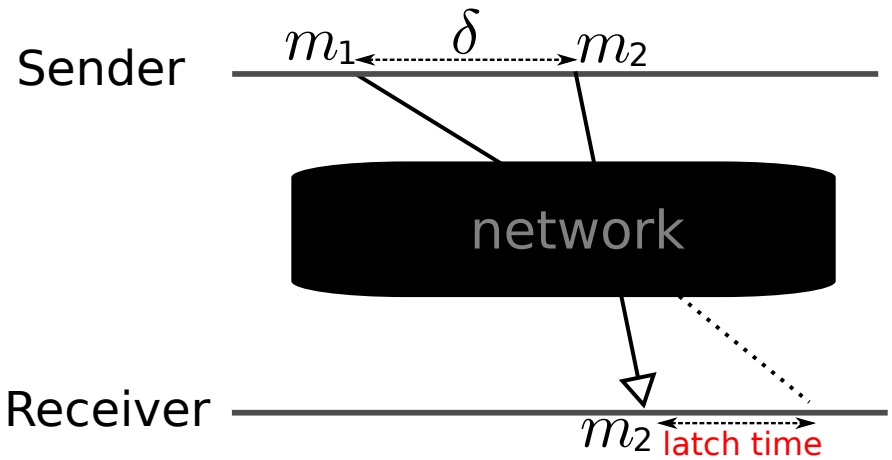
Latch Time



- Is m_1 relevant (matching) ?
- Let's assume it is...

- We would like Pr(FIFO violation) $\leq P_t$ ($P_t = 0.05$)

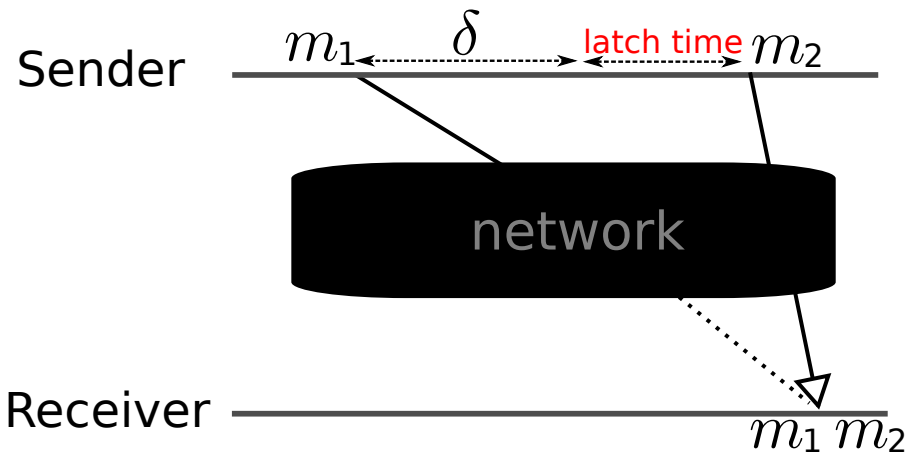
Latch Time



- Is m_1 relevant (matching) ?
- Let's assume it is...

- We would like Pr(FIFO violation) $\leq P_t$ ($P_t = 0.05$)
- Latch m_2 *hoping* for m_1 to arrive

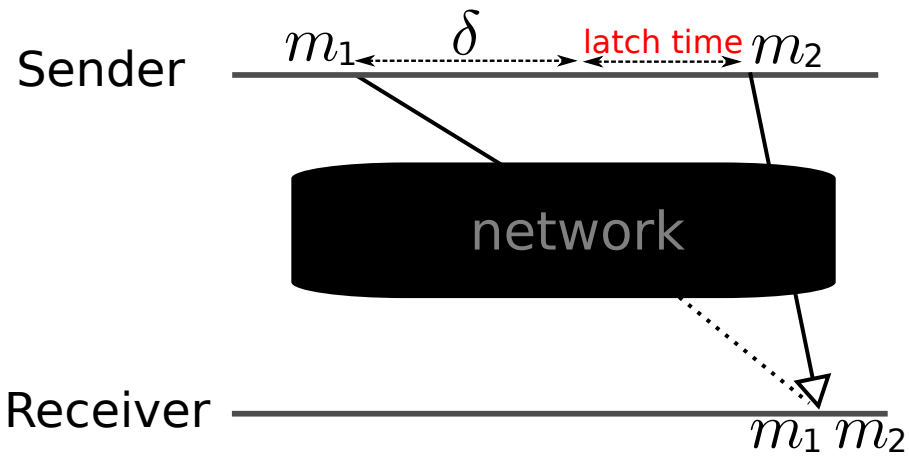
Latch Time



- Is m_1 relevant (matching) ?
- Let's assume it is...

- We would like Pr(FIFO violation) $\leq P_t$ ($P_t = 0.05$)
- Latch m_2 *hoping* for m_1 to arrive
- Pr(FIFO violation) $<$ Pr($delay(m_1) - delay(m_2) > \delta + \tau$) $< P_t$

Latch Time



- Is m_1 relevant (matching) ?
- Let's assume it is...

- We would like Pr(FIFO violation) $\leq P_t$ ($P_t = 0.05$)
- Latch m_2 *hoping* for m_1 to arrive
- Pr(FIFO violation) $<$ Pr($delay(m_1) - delay(m_2) > \delta + \tau$) $< P_t$
- τ can be found using probabilistic inequalities or the Hypoexponential distribution

Publication Record

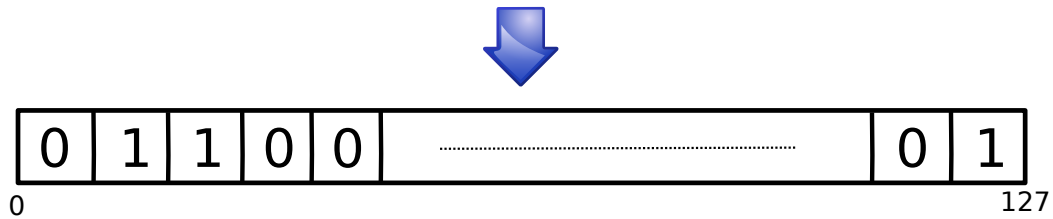
Publication Record

- Carzaniga et al. proposed an encoding scheme that encodes messages and subscriptions in Bloom filters.

Publication Record

- Carzaniga et al. proposed an encoding scheme that encodes messages and subscriptions in Bloom filters.

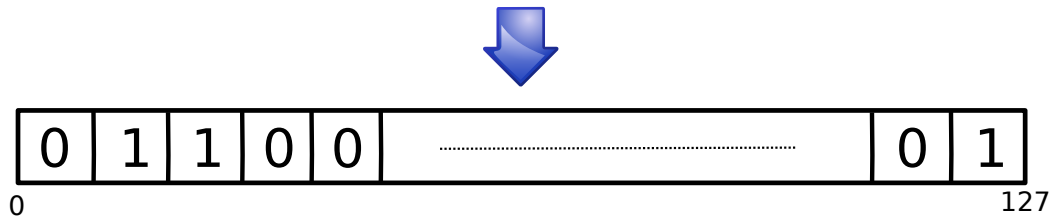
(class = "stock'', symbol = "IBM'', price =301)



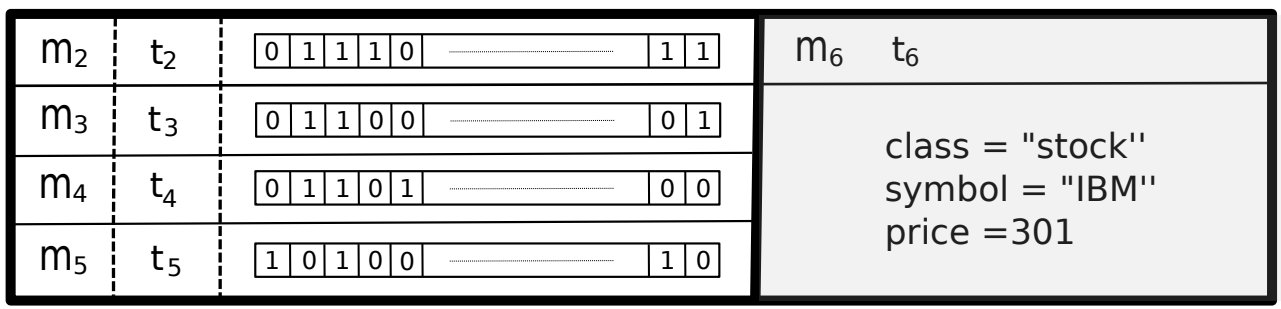
Publication Record

- Carzaniga et al. proposed an encoding scheme that encodes messages and subscriptions in Bloom filters.

(class = "stock", symbol = "IBM", price =301)



- Sender piggybacks each message with encoded summary of last k messages (**publication record**)



Publication Record

Publication Record

- Using publication record a recipient can decide if a missing message is of interest or not (for a limited number of missing messages)

Publication Record

- Using publication record a recipient can decide if a missing message is of interest or not (for a limited number of missing messages)
- So if a missing message is not relevant we just don't wait for it

Reliability

Reliability

- By reliability we mean going beyond best effort delivery

Reliability

- By reliability we mean going beyond best effort delivery
- Reliable IP multicast has been studies before:
  - Reliable Multicast Protocol (RMTP) (IETF Standard)
  - Pragmatic General Multicast (PGM) (IETF Draft)
  - Scalable Reliable Multicast (SRM)
    - ▷ **Cooperative message recovery**

Reliability

- By reliability we mean going beyond best effort delivery
- Reliable IP multicast has been studies before:
  - Reliable Multicast Protocol (RMTP) (IETF Standard)
  - Pragmatic General Multicast (PGM) (IETF Draft)
  - Scalable Reliable Multicast (SRM)
    - ▷ **Cooperative message recovery**
- But again...pub/sub has its own particularities

Reliability

- Two challenges:

Reliability

- Two challenges:
- How to detect a missing message ?
  - ▷ Normally we would use sequence numbers (SRM)
  - ▷ Using **publication record**

Reliability

- Two challenges:
- How to detect a missing message ?
  - ▷ Normally we would use sequence numbers (SRM)
  - ▷ Using **publication record**
- How to retrieve a missing message ?
  - ▷ In SRM we would multicast a request to the multicast group
  - ▷ No such thing as **multicast group**

Reliability

- Two challenges:
- How to detect a missing message ?
  - ▷ Normally we would use sequence numbers (SRM)
  - ▷ Using **publication record**
- How to retrieve a missing message ?
  - ▷ In SRM we would multicast a request to the multicast group
  - ▷ No such thing as **multicast group**
- Using pub/sub API !

Loss Recovery

- Volunteers subscribe to request messages
  - ▷ s=$\{$*reliability-message*=*request*, *message*=*msg_id*$\}$

Loss Recovery

- Volunteers subscribe to request messages
  - ▷ s=$\{$*reliability-message*=*request*, *message*=*msg_id*$\}$
- A request is a published message
  - ▷ r=$\{$*reliability-message*=*request*, *message*=*msg_id*$\}$

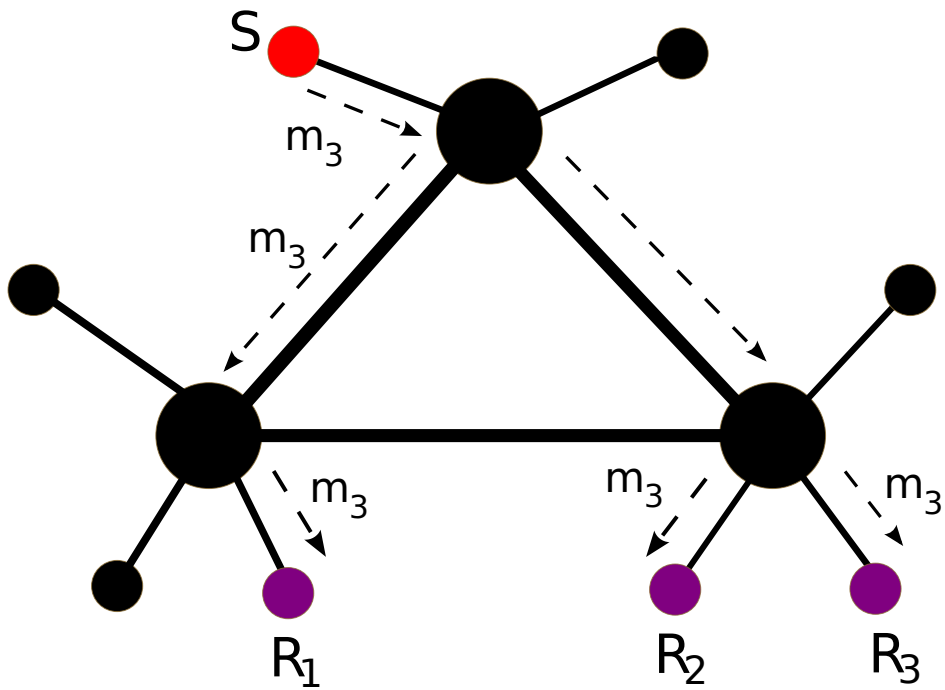
Loss Recovery

- Volunteers subscribe to request messages
  ▷ s=$\{$*reliability-message*=*request*, *message*=*msg_id*$\}$
- A request is a published message
  ▷ r=$\{$*reliability-message*=*request*, *message*=*msg_id*$\}$
- But this is too coarse grained

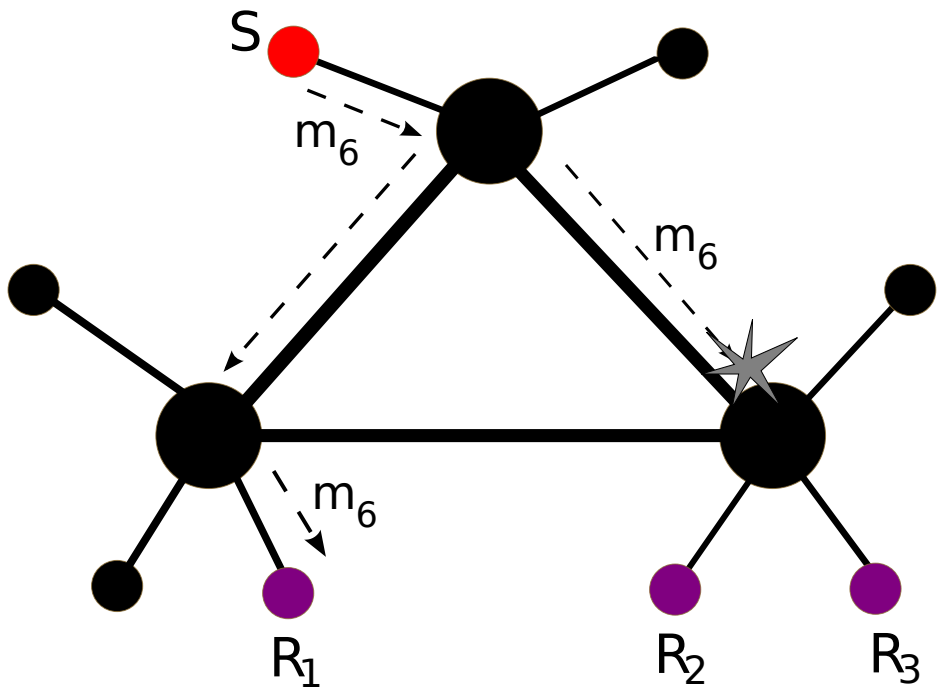
Loss Recovery

- Volunteers subscribe to request messages
  ▷ s={*reliability-message*=*request*, *message*=*msg_id*}
- A request is a published message
  ▷ r={*reliability-message*=*request*, *message*=*msg_id*}
- But this is too coarse grained
- We further enhance request routing with Bloom filters
  ▷ Additional constraint/attribute that give hints about the content of messages

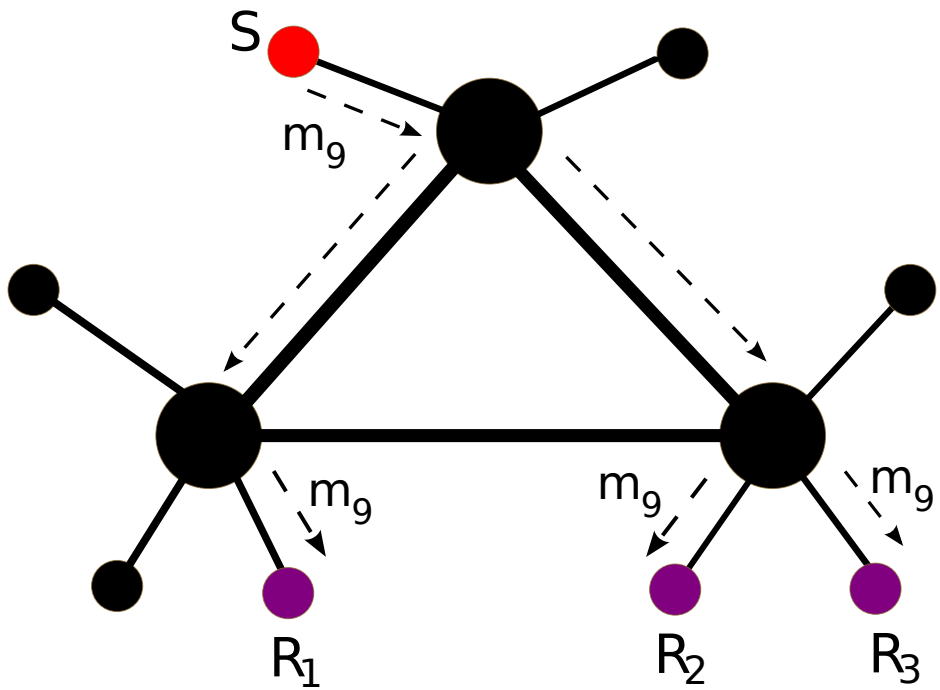
Reliability: Example



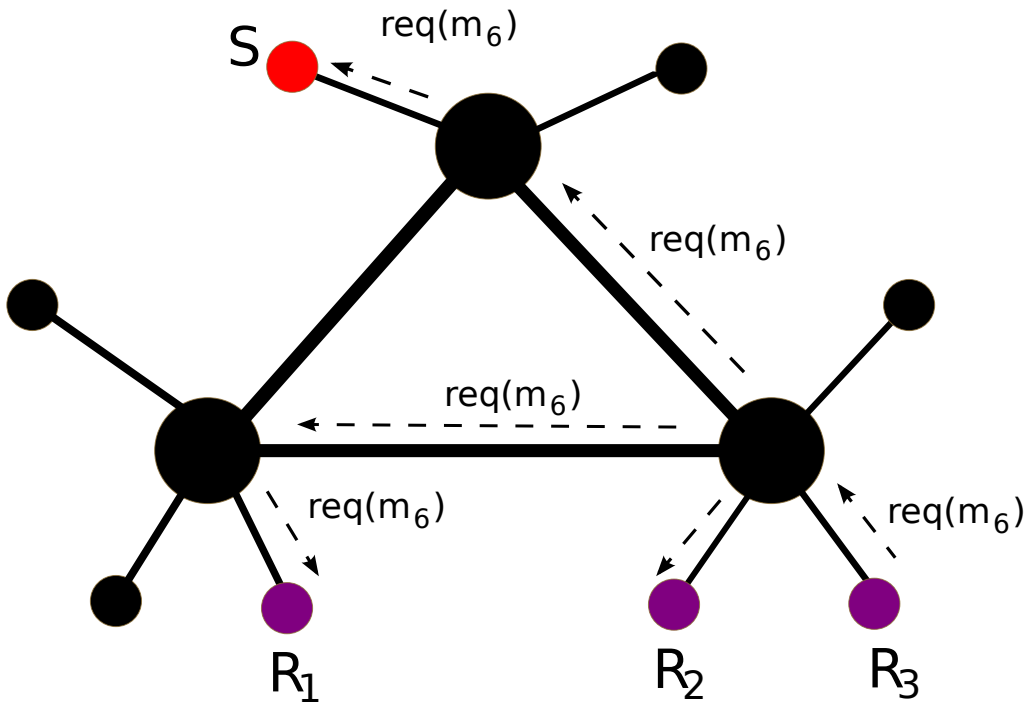
Reliability: Example



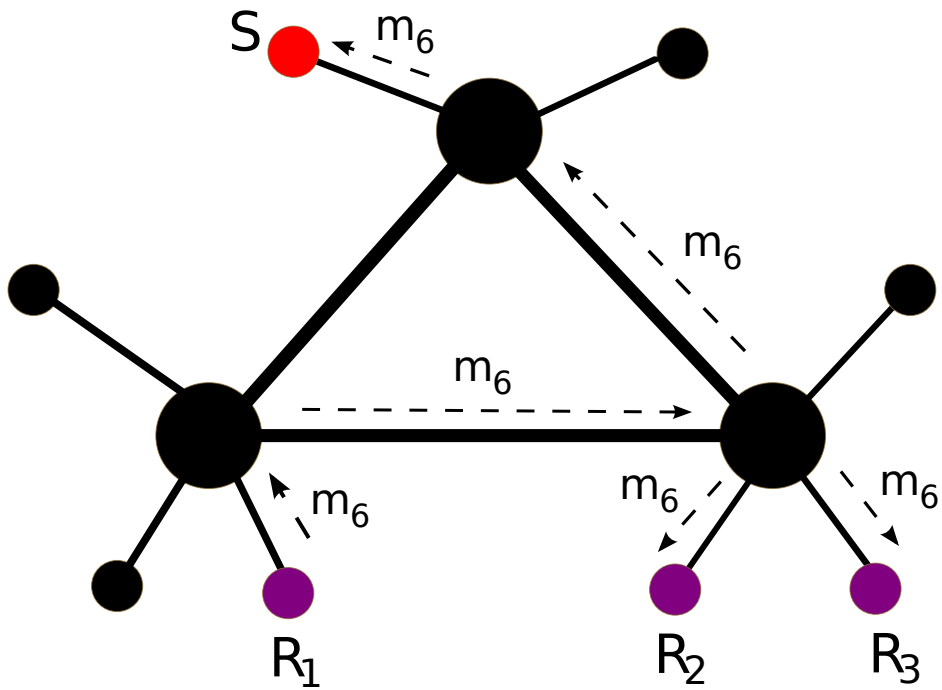
Reliability: Example



Reliability: Example



Reliability: Example



Implementation and Evaluation

- A pluggable module written in Java
  - ▷ Integrates into any pub/sub system

Implementation and Evaluation

- A pluggable module written in Java
  - ▷ Integrates into any pub/sub system
- With Siena B-DRP mitigated up to 99.5% of FIFO violations

Implementation and Evaluation

- A pluggable module written in Java
  - ▷ Integrates into any pub/sub system
- With Siena B-DRP mitigated up to 99.5% of FIFO violations
- Recovered more than 70% of messages losses in very unstable networks

Implementation and Evaluation

- A pluggable module written in Java
  - ▷ Integrates into any pub/sub system
- With Siena B-DRP mitigated up to 99.5% of FIFO violations
- Recovered more than 70% of messages losses in very unstable networks
- With a small sacrifice of throughput and end-to-end delay

Future Directions

- Theoretical analysis of the protocol performance
- Better encoding schemes for publication record
  - ▷ Exploiting temporal locality of events

Thank you!

Questions?